Security Games with Interval Uncertainty

Submission 263

ABSTRACT

Security games provide a framework for allocating limited security resources in adversarial domains, and are currently used in applications including security at the LAX airport, scheduling for the Federal Air Marshals, and patrolling strategies for the U.S. Coast Guard. One of the major challenges in security games is finding solutions that are robust to uncertainty about the game model. Bayesian game models have been developed to model uncertainty, but algorithms for these games do not scale well enough for many applications, and the problem is NP hard.

We take an alternative approach based on using intervals to model uncertainty in security games. We present a fast polynomial time algorithm for security games with interval uncertainty. This provides the first viable approach for computing robust solutions to very large security games. In addition, we introduce a methodology for approximating the solutions to infinite Bayesian games with distributional uncertainty using intervals to approximate the distributions. We show empirically that using intervals is an effective approach for approximating solutions to these Bayesian games; our algorithm is both faster and results in higher quality solutions than the best previous methods.

1. INTRODUCTION

Security games [11, 21] are a general framework for modeling a wide variety of resource allocation decisions in adversarial security domains. These games are used to find optimal randomized strategies for a defender to deploy limited security resources to protect vulnerable targets from attacks. Recently, they have been used in a growing number of homeland security applications, including airport security [18, 19], scheduling for the Federal Air Marshals [22], and patrolling strategies for the United States Coast Guard [20]. Game theory is also increasingly used for applications in cybersecurity [3, 15].

An important concern with using game theory for modeling real-world security problems is that the game models require very precise and accurate information about the capabilities and preferences of the players as inputs. In practice, models are constructed using information provided by subject matter experts knowledgeable about the resources available for protection, the security risks and vulnerabilities of different targets, and the motivations of possible attackers. While this information may be available, there is a high degree of uncertainty associated with some of the key modeling parameters. For example, it is difficult to know exactly what value a terrorist might perceive for a successful attack against a given target, even through it may be clear that some targets are considerably more attractive than others. This means that it may not be possible to give exact values for the payoffs in different attack scenarios in a security game.

There is a growing emphasis on developing models and algorithms for security games that are able to represent various kinds of uncertainty about the model, with the ultimate goal of generating robust security strategies that are not highly sensitive to modeling errors. The existing approaches are primarily based on Bayesian Stackelberg game formulations that model uncertainty about payoffs, the observation capabilities of an attacker, and other factors [16, 17, 10, 12, 25]. All of these current approaches suffer from problems with computational scalability and/or solution quality. Finite Bayesian Stackelberg games are NP hard to solve [6] in theory, and are computationally challenging in practice as well. There are no exact algorithms for infinite Bayesian Stackelberg games, and all of the existing solution methods lack guarantees on solution quality [12].

The approach we take in this work is based on modeling uncertainty in security games using intervals, rather than distributions. We take a worst-case optimization approach with respect to the interval uncertainty. In our model, the defender in a security game knows only that the attackers payoffs fall within some interval of possible values, and tries to maximize the worst case outcome for any possible realization of payoff values consistent with these intervals. Modeling uncertainty using intervals is common in the literature on robust optimization [5], and this basic idea has also been extended to develop a notion of equilibrium in game theory based on robust optimization [1]. The most closely related model in security games is the BRASS model introduced by Pita et al. for robustness against human decision-makers [17], which is a special case of our model.

We show in this work that the interval-based approach has advantages over Bayesian approaches for modeling uncertainty in security games. It is much simpler for domain experts to understand and specify a model based on interval uncertainty because the model does not require detailed information about probability distributions which is problematic to elicit. In many cases, an interval model is a more natural and effective way to represent the game model. We show in this paper that interval models also have considerable computational advantages over Bayesian models. While Bayesian models are NP-hard, we introduce a polynomial-time approximation algorithm that is scalable and provides tight bounds on solution quality. For large security games, our algorithm based on interval uncertainty may be the only computationally feasible

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

approach for handling uncertainty.

Specifically, in this paper we make the following primary contributions: (1) we introduce an interval-based model of uncertainty for security games, (2) we present a very fast polynomial-time algorithm for solving interval security games, (3) we present a methodology for approximating security game with distributional uncertainty using intervals, which can be solved using our algorithm, and (4) we present experimental results showing the value of the interval method for increasing robustness, and showing that interval-based methods can also provide fast approximations with high solution quality even when distributional information is available.

2. RELATED WORK

One of the key motivations for this work is the important applications of Stackelberg games to real-world security domains [21]. They have been used in fielded applications at the Los Angeles International Airport [18], the Federal Air Marshals Service [22], for the Coast Guard [20], and in other areas. There is also work on using game theory for patrolling strategies for robots and unmanned vehicles [7, 2, 4] and applications of game theory in network security [3, 24, 15]. Much of this work is computational in nature, and progress on security games has been driven by many algorithmic advances that allow use to solve ever larger and more complex games [6, 16, 9].

Robustness and uncertainty have been important concerns in game theory since very early. One of the most influential models, Bayesian games, was developed early in the history of game theory by John Harsanyi [8]. Worst-case approaches also have a long history in game theory, including many approaches that build on the pessimistic notion of minmax strategies. However, many of these take a worst-case view with respect to opponent behaviors, rather than the underlying game model. One recent exception is the notion of robust equilibrium [1], which takes a worst-case approach inspired by the robust optimization literature. There are also several existing works that look at varying aspects of the problem of robustness in security games, most of which adopt a Bayesian framework for reasoning about uncertainty [16, 17, 10, 12, 25].

3. SECURITY GAMES WITH INTERVALS

We first introduce the basic security game model [11], and then extend this model to include interval uncertainty about the attacker's payoffs. A security game has two players, a *defender*, Θ, and an *attacker*, Ψ. The defender is protecting a set of *targets* $T = \{t_1, \ldots, t_n\}$ (e.g., airport terminals or computer servers) from attacks using a limited number of resources, with the number of available resources denoted by m. We assume that all resources are identical and can be used to protect any target. The set of pure strategies for the attacker, denoted $\sigma_\Psi \in \Sigma_\Psi$, correspond to actions attacking exactly one target from T. Each pure strategy for the defender, denoted $\sigma_\Theta \in \Sigma_\Theta$, corresponds to a subset of targets from T with size less than or equal to m. These are the targets that the defender chooses to protect.

Following previous work on security games, we model the interaction as a Stackelberg game [23]. The defender first commits to a mixed strategy δ_Θ that is a probability distribution over the pure strategies from Σ_Θ. The attacker then observes this mixed strategy δ_Θ, and chooses a best response strategy from Σ_Ψ that gives the attacker the maximum possible payoff. As in the previous work on security games [21], we use Strong Stackelberg Equilbrium as the standard solution concept, so ties in the attacker's best responses are broken in favor of the defender. In addition, we only need to consider pure strategies for the attacker, since there will always exist a pure-strategy best response [16].

The payoffs for the game depend on which of the n targets is attacked, and whether or not the target is protected (covered) by the defender or not. Specifically, for an attack on target t, the defender receives a payoff $U^u_\Theta(t)$ if the target is uncovered, and $U^c_\Theta(t)$ if the target is covered. The payoffs for an attacker of type $\omega \in \Omega$ is $U^u_\Psi(t, \omega)$ for an attack on an uncovered target, and $U^c_\Psi(t, \omega)$ for an attack on a covered target. We say that an attack on a covered target is "unsuccessful" and an attack on an uncovered target is "successful." In a security game we also assume that $U^c_\Theta(t) \geq U^u_\Theta(t)$ and $U^u_\Psi(t) \geq U^c_\Psi(t)$ for all $t \in T$. In games with identical and unconstrained defender resources, we can use a compact representation for the defenders strategies [11]. We represent the defender strategies as *coverage vectors* which give the probability that there is a defender resource assigned to each target. These probabilities for each target t_i are denoted by c_i, with $\sum_{i=1}^{n} c_i = m$, and the full vector of probabilities is denoted by C.

Extension to Interval Uncertainty. We now introduce the *Interval Security Games* (ISG) model, which extends the standard security game model so that the defender has uncertainty about the attacker's payoffs in the form of intervals. We still assume that both the attacker and defender know their *own* payoffs with certainty. In addition, we do not need to model the attacker as having uncertainty about the defender's payoffs because the attacker is able to directly observer the strategy of the defender, and therefore does not require knowledge of the payoffs to predict the defender's strategy.

In the model, rather than having a single value representing the attacker's payoffs for the two cases ($U^u_\Psi(t)$ and $U^c_\Psi(t)$), we have pairs of values that represent the maximum and minimum possible payoffs for the case of a successful or unsuccessful attack on target t_i. We denote these values using the notation $U^{u,max}_\Psi(t)$ and $U^{u,min}_\Psi(t)$ for the uncovered case, and $U^{c,max}_\Psi(t)$ and $U^{c,min}_\Psi(t)$ for the covered case. The idea is that the defender knows only that the attacker's payoffs lie within some possible range of values, and not the precise value. The defender does not have information about the distribution of payoffs within these intervals, and therefore cannot compute an expected payoff, so we cannot use the standard concept of Strong Stackelberg Equilbrium. Instead, we follow the literature on robust optimization and take a worst-case approach. The defender's goal in our framework is to select a coverage vector, C, that maximizes the defenders worst-case payoff over all of the possible ways that the attacker payoffs could be chosen from the defined intervals.

4. ANALYSIS OF ISG

In security games without intervals, we define the *attack set* to be the set of all targets that give the attacker the maximum expected payoff, given some coverage strategy C. For some classes of security games, finding the optimal coverage strategy can be reduced to finding a coverage strategy that induces the maximum attack set, while minimizing the attacker's expected payoff. This result is the basis of the ORIGAMI algorithm [11].

In our model we cannot directly apply the idea of the attack set, but we can generalize this idea as follows. We define the *potential attack set* for a coverage strategy C to be the set of all targets that *could* give the attacker the maximum expected value, for any realization of attacker payoffs consistent with the payoff intervals. For every target, the attacker has a range of expected payoffs:

$$v^{max}(t_i) = c_i \cdot U^{c,max}_\Psi(t_i) + (1 - c_i) \cdot U^{u,max}_\Psi(t_i) \qquad (1)$$

$$v^{min}(t_i) = c_i \cdot U^{c,min}_\Psi(t_i) + (1 - c_i) \cdot U^{u,min}_\Psi(t_i). \qquad (2)$$

Observe that for a given coverage vector C the attacker can guarantee a payoff of at least the maximum of the minimum values over all targets; let us denote this value by $R = \max_{t_i} v^{min}(t_i)$. Given the value of R we can identify the targets that could be attacked for some realization of the payoff values. Any target t_i with a maximum expected value $v^{max}(t_i) \geq R$ could be the best target for the attacker to attack. To see this, suppose that the the attacker's payoff for t_i is the maximum value in the interval, and the payoffs for all other targets is are the minimal values, so that the best possible value for attacking any target other than t_i is R. Therefore, the potential attack set, $\Lambda(C)$, is defined as:

$$\Lambda(C) = \{t_i : v^{max}(t_i) \geq R\} \tag{3}$$

The defender's expected payoff for each target is:

$$d_i = c_i \cdot U^c_\Theta(t_i) + (1 - c_i) \cdot U^u_\Theta(t_i). \tag{4}$$

The defender's objective is to select a strategy C to maximize the worst-case payoff over all of the targets in the potential attack set:

$$\max_C (\min_{t_i \in \Lambda(C)} d_i) \tag{5}$$

This problem cannot be solved using linear programming because the set of targets $t_i \in \Lambda(C)$ depends on C. It can be expressed as a mixed-integer program (MIP) which is a slightly generalized version of the MIP used for BRASS [17]. We omit this MIP due to space constraints, but it can be found in [17].

The main idea of our approach for finding a coverage strategy with the desired properties is to transform this optimization problem into a series of feasibility problems. Our first observation is that the defender's maximum possible expected payoff increases monotonically as the number of available resources m is increased. This follows as direct consequence of the fact that the defender's set of possible coverage strategies is strictly larger for larger m.

Using this observation, we can frame the problem as a binary search problem in the space of defender expected payoffs. At each iteration, we will test whether the defender payoff at the midpoint is feasible or not given the number of resources available. If it is not, the maximum payoff must be smaller than the midpoint, if it is feasible, the maximum payoff is greater than or equal to the midpoint. Using this strategy, we can approximate the maximum payoff to within a very small tolerance.

To implement this approach, we need to analyze the problem to determine whether a given defender payoff which we denote as D^* is feasible given the resources available, m. Since we are interested in worst-case outcomes, this means that we need to guaranteed that the defender will achieve *at least* D^* for any attacker payoffs in the known intervals. For D^* to be guaranteed by a particular coverage strategy C, one of the following two conditions must hold for every target:

1. The target is in the potential attack set $\Lambda(C)$, but the defender's expected payoff for attacking the target is greater than D^*
2. The target is not in the potential attack set $\Lambda(C)$.

We will now derive conditions that satisfy these conditions for each target using the minimum amount of resources (i.e., coverage probability). We can calculate the coverage required on each target to satisfy condition 1 for each target (if it is in Λ) from the equation for the defender's payoff. The minimal coverage for target t_i is given by:

$$c^1_i = \max(0, 1 - \frac{D^* - U^u_\Theta(t_i)}{U^c_\Theta(t_i) - U^u_\Theta(t_i)}). \tag{6}$$

The problem now reduces to finding the potential attack set that will minimize the overall coverage probability required to meet conditions 1 and 2 for all targets, since pushing some A naïve approach would be to enumerate all of the possible attack sets and calculate the minimum coverage for each such set. For any given set, we can calculate the value of R directly, and then calculate the minimal coverage required for each target in Λ from Equation 6. For a given value of R we can also calculate the minimum coverage that would be required on each target t_i so that the target is *not* in Λ, which requires that the maximum possible expected payoff for the attacking t_i is less than R. We calculate the minimum coverage as follows, using the maximum attacker payoffs from the possible intervals:

$$c^2_i = \max(0, 1 - \frac{R - U^{u,max}_\Psi(t_i)}{U^{c,max}_\Psi(t_i) - U^{u,max}_\Psi(t_i)}). \tag{7}$$

By summing the values of c^1_i for targets in Λ and c^2_i for the remaining targets, we get the minimum coverage required to guarantee D^* for this potential attack set. Unfortunately, the number of such sets is exponential in the number of targets, so this direct approach is highly inefficient. To avoid this problem we make another observation that allows us to more efficiently explore candidate solutions. For every possible set Λ there must be some target, which we label $\hat{t}$ that has the maximum minimum expected payoff, R. This is the target that defines the value of R. Since there are only n targets, we can test each target as a candidate for $\hat{t}$, and construct a coverage vector that meets the necessary constraints using minimal resources under this assumption. We present further the details of this construction in the next section. If the solution is feasible for any one of the n targets that are candidates for $\hat{t}$, then the value of D^* is feasible. In the following section we describe an algorithm that uses this solution strategy to efficiently approximate the optimal coverage vector C for the defender.

5. ISG ALGORITHM

We now describe our algorithm for solving interval security games, which we refer to as the ISG solver. The pseudocode is given in Algorithms 1 and 2. Algorithm 1 implementes binary search in the space of possible defender payoffs, as described in the previous section. The feasibility check is presented in Algorithm 2, based on the analysis presented above.

The goal is to construct a solution that will guarantee the defender D^* while using the minimum resources; if we can construct a solution that uses less than the available resources m we have found a feasible solution. The strategy is to divide the search into n possible cases, each of which corresponds to a different assumption about which of the targets will have the maximum minimum expected payoff, R. The algorithm iterates through each possible choice of t_i as a candidate for this special target $\hat{t}$. For each of these cases the algorithm constructs a coverage vector using minimal coverage probability that guarantees the defender D^* based on the conditions 1 and 2 above.

First, for $\hat{t}$ the target is part of the potential attack set in this solution based on our assumption that it will have the maximum minimum expected payoff. Therefore, if this target is attacked it must give the defender an expected payoff of at least D^*, as calculated in Equation 6. We take the value $c^1_{\hat{t}}$ necessary to ensure D^* and use this to calculate the value of R. This value of R is as high as possible because we use the minimum coverage. We do not need to consider adding additional coverage to $\hat{t}$ to decrease the value of R because this could only increase the coverage needed for any other target. To see this, note that the values of c^1_i are independent of R

and the values of c_i^2 increase monotonically as R decreases.

Now that we have the value of R, we calculate the value of c_i^2 for every other target. We can then calculate the minimum coverage required to satisfy one of these two conditions by taking $\min(c_i^1, c_i^2)$ for each target. There is one final condition that must be met for each target for our initial assumption to hold–it must be the case that the target we are assuming is $\hat{t}$ actually has the maximum minimum expected payoff. We guarantee this by adding one additional constraint on the coverage probability assigned to each target. This constraint states that the maximum attacker payoff for the target is less than the calculated value of R for $\hat{t}$:

$$c_i^3 = \max(0, 1 - \frac{R - U_\Psi^{u,min}(t_i)}{U_\Psi^{c,min}(t_i) - U_\Psi^{u,min}(t_i)}). \qquad (8)$$

These values also increase monotonically as R decreases. The final calculation for the minimum coverage that must be placed on each target is $\max(c_i^3, \min(c_i^1, c_i^2))$. We sum these coverages over all targets and compare this with the available resources m to determine whether this selection of $\hat{t}$ yields a feasible solution. If this selection does not yield a feasible solution, the algorithm continues testing the other possible targets as selections for $\hat{t}$. As soon as a feasible solution is found, the subroutine terminates and the binary search continues.

The worst-case complexity of the algorithm is $O(n^2 \cdot \log(1/\epsilon))$ where ϵ is the error tolerance parameter for the binary search. Each feasibility check requires one iteration to test each target as $\hat{t}$, and each iteration does several constant-time operations on each target to determine the minimal coverage. Therefore, the feasibility check has complexity $O(n^2)$. Binary search requires $O(\log(1/\epsilon))$ iterations to converge within ϵ, giving the overall complexity of $O(n^2 \cdot \log(1/\epsilon))$.

Algorithm 1 ISG Solver

```
for all $t_i \in T$ do
  $c_i \leftarrow 0$
end for
$maxPayoff \leftarrow 0$
$minPayoff \leftarrow \min_{t_i \in T} U_\Theta^u(t_i)$
while $maxPayoff - minPayoff > \epsilon$ do
  $midPoint \leftarrow (maxPayoff - minPayoff)/2$
  if feasibilityCheck($midPoint$, $m$, $C$) then
    $minPayoff \leftarrow midPoint$
  else
    $maxPayoff \leftarrow MidPoint$
  end if
end while
return $C$
```

In addition to ISG we implemented a mixed-integer program that computes an exact solution for our interval security games. This MIP model is used as a benchmark in the experimental evaluation. We do not described this MIP in detail here due to space constraints, but note that the formulation is a minor generalization of the BRASS MIP formulation presented in Pita et al. [17].

6. SECURITY GAMES WITH DISTRIBUTIONAL UNCERTAINTY

An alternative way to model uncertainty about payoffs in security games is to use distributions instead of intervals to represent possible values. The distributional security games (DSG) model introduced by Kiekintveld et al. [12] uses this approach, and presents a variety of approximation algorithms for computing solutions to the resulting infinite Bayesian Stackelberg games. The DSG model contains more information than our model because it has access to distributional information, however, this has two significant drawbacks:

1. The models are more problematic to accurately define, since they require specifying many payoff distributions which requires a large amount of very precise information from the modeler.

2. It is computationally very challenging to compute optimal solutions to an infinite Bayesian Stackelberg game; no exact algorithms are known, and even heuristic approximations are computationally expensive.

For these reasons, it may often be preferable to directly use an interval model instead of a distributional model. However, we also show in this paper that we can use our interval algorithm as an efficient way to approximate high-quality solutions to a distributional model. Here, we briefly introduce the DSG model, and then show how we can transform a distributional game into an interval game which approximates the distributional game, which can then be solved with our interval algorithm. In the experimental results we compare this approximation using intervals with the best known approximation methods which operate directly on the distributional model using Monte Carlo sampling.

A *disributional security game* (DSG) extends the standard security game model from Section 3 in a similar way to our interval security game model. The difference is that the attackers payoffs are represented by continuous probability density functions (e.g., uniform distributions or Gaussian distributions) instead of intervals. Formally, this becomes an infinite Bayesian Stackelberg game with an infinite number of attacker types, and the game unfolds as

Algorithm 2 feasibilityCheck

```
for all $t_i \in T$ do
  $c_i^1 \leftarrow \max(0, 1 - \frac{midPoint - U_\Theta^u(t_i)}{U_\Theta^c(t_i) - U_\Theta^u(t_i)})$
end for
for all $t_i \in T$ do
  $totalCov \leftarrow c_i^1$
  $c_i \leftarrow c_i^1$
  if $c_i > 1$ then
    $GOTO\ next\ t_i$
  end if
  $R \leftarrow (c_i^1 \cdot U_\Psi^{c,min}(t_i)) + ((1 - c_i^1) \cdot U_\Psi^{u,min}(t_i)) - \epsilon'$
  for all $t_j \in \{T \backslash t_i\}$ do
    $c_j^2 \leftarrow \max(0, 1 - \frac{R - U_\Psi^{u,max}(t_j)}{U_\Psi^{c,max}(t_j) - U_\Psi^{u,max}(t_j)})$
    $c_j^3 \leftarrow \max(0, 1 - \frac{R - U_\Psi^{u,min}(t_j)}{U_\Psi^{c,min}(t_j) - U_\Psi^{u,min}(t_j)})$
    $minCov \leftarrow max(c_i^3, \min(c_i^1, c_i^2))$
    if $minCov < 0$ || $minCov > 1$ then
      $GOTO\ next\ t_i$
    end if
    $totalCov \leftarrow totalCov + minCov$
    $c_j \leftarrow minCov$
  end for
  if $totalCov \leq m$ then
    return TRUE, $C$
  end if
end for
return FALSE
```

follows: (1) the defender commits to a mixed strategy (2) nature chooses a random attacker type $\omega \in \Omega$ with probability $Pb(\omega)$, (3) the attacker observes the defender's mixed strategy, and (4) the attacker responds to the mixed strategy with a best-response that provides the attacker (of type ω) with the highest *expected* payoff. We define the type distribution by replacing the payoffs values $U_\Psi^c(t,\omega)$, $U_\Psi^u(t,\omega)$ for each target $t \in T$ with two continuous probability density functions:

$$f_\Psi^c(t,r) = \int_{\omega\in\Omega} Pb(\omega)U_\Psi^c(t,\omega)d\omega \tag{9}$$

$$f_\Psi^u(t,r) = \int_{\omega\in\Omega} Pb(\omega)U_\Psi^u(t,\omega)d\omega \tag{10}$$

that represent the defender's *beliefs* about the attacker payoffs. For example, the defender expects with probability $f_\Psi^c(t,r)$ that the attacker receives payoff r for attacking target t when it is covered. For some coverage vector C, let $X_t(C)$ be a random variable that describes the *expected* attacker payoffs for attacking target t, given C. We then define the probability that the attacker will choose target t for each target $t \in T$ as follows:

$$a_t(C) \quad = \quad Pb[X_t(C) > X_{t'}(C) \text{ for all } t' \in T \setminus t] \tag{11}$$

because the attacker acts rationally. Conceptually, this gives the probability that the attacker will choose to attack each target for a given coverage vector C and the probability distributions of the attacker's payoffs. Using these probabilities, we can calculate the expected payoff for the defender. The original paper presents a derivation of an analytic formula for these probabilities, but it cannot be solved directly. Instead, Monte Carlo simulation is used to estimate the attack probabilities. We generate one sample attacker type by sampling payoffs from each of the payoff distribution; these are the payoff values assigned to that type. Using those payoff values we can calculate the best-response for this attacker type against the coverage strategy C. Sampling a large number of types in this way is used to estimate the expected value of a coverage strategy for a DSG.

In our experimental results, we benchmark again the Greedy Monte Carlo (GMC) algorithm introduced by Kiekintveld et al. This was found to be the method that was fastest and gave the highest quality solutions for instances of DSG, especially when scaling up to large games. The GMC method is based on sampling a large number of attacker types (often thousands) using Monte-Carlo sampling. It then uses a greedy heuristic to approximate the optimal defender coverage strategy against the sampled attacker types.

To apply our interval algorithm to distributional security games we translate the distribution for each payoff to an interval. There are many possible ways to do this but we use a simple method that centers the interval around the mean of the distribution and determines the size of the interval based on the standard deviation of the interval and a multiplier. The multiplier is a parameter of the algorithm, and allows us to have intervals that include a larger or smaller fraction of the possible payoff values in the distribution. The minimum value for the interval is calculated as $mean - (StdDev \cdot multiplier)$ and the maximum values is calculated as $mean + (StdDev \cdot multiplier)$. Figure 1 shows this visually.

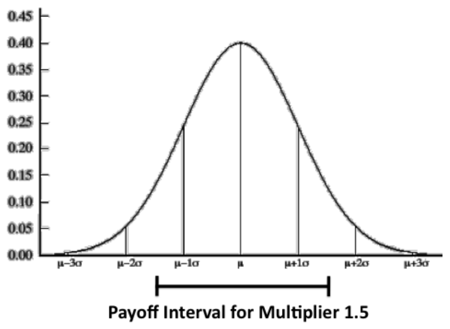


Figure 1: A payoff interval for a Gaussian distribution.

7. EXPERIMENTAL EVALUATION

We begin by evaluating the runtime and solution quality of the ISG solver on interval security games, and then present results showing the performance of using this method to solve distributional security games.

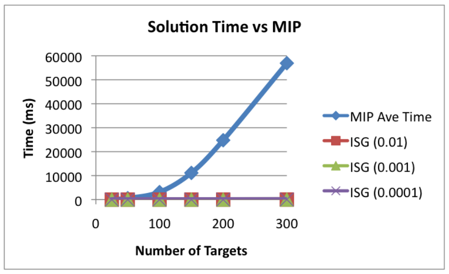
7.1 Interval Game Experiments

First, we tested the speed of the ISG solver against an exact MIP formulation based on BRASS. We tested the algorithms on 30 randomly generated sample games with defender payoffs for successful attacks drawn uniform random between 0 and -100, and the attacker payoffs for successful attacks drawn uniformly between 0 and 100. We modify the attacker payoffs to be intervals by using the first value drawn as the minimum value and setting the maximum value by adding a uniform random value between 0 and 20. The payoffs for unsuccessful attacks in this experiment were 0 for both players. We fixed the number of resources at 20% of the number of targets.

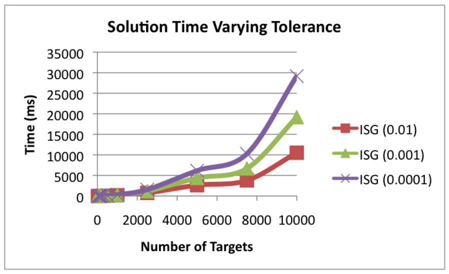
In Figure 2(a) we present results for the MIP (solved using GLPK version 4.36) and ISG with three different tolerance settings. The tolerance settings control the accuracy of the binary search. All three ISG instances are much faster, even with an error tolerance of just 0.0001. Figure 2(b) shows results for the three ISG settings on much larger games. Here we see a modest increase in solution time with increasing accuracy. Even for 10000 targets and the highest accuracy setting, ISG solves the game in half the time required by the MIP to solve games with only 300 targets.

We also ran some experiments looking at the impact of interval uncertainty on solution quality under varying assumptions about the attacker strategy. For this experiment we used 200 randomly generated games with defender payoffs for successful attacks drawn from the range -20 to -10, and the attacker payoffs in the range 10 to 20. Payoffs for unsuccessful attacks were 0 for both players. The baseline case has no uncertainty about the attacker's payoffs. We solved these games using an existing linear-time solver for security games [14]. We then added increasing amounts of interval uncertainty to the attacker's payoffs, and solved the resulting games using the ISG solver.

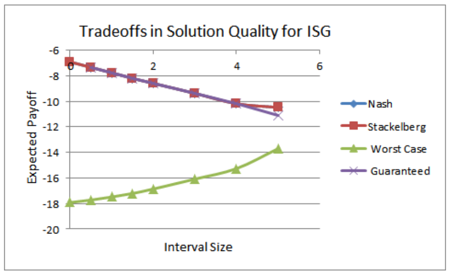
The results are shown in Figure 2(c). On the x-axis is the size of the intervals for the attacker's payoffs. On the y-axis is the expected payoff for the defender. The four lines represent four different assumptions about the attacker. The Nash attacker always plays the optimal attacker strategy computed in the case with no uncertainty (in this case, the Stackelberg equilibrium strategy is the same as the Nash strategy [13]). The Stackelberg attacker is able to observe the exact coverage strategy used in each case, and chooses a best-response, as in a Strong Stackelberg Equilibrium. The worst case attacker always chooses the worst possible target for the defender, without regard to the attacker's own payoffs. Finally, the Guaran-



(a) Comparison of solution time for ISG and the MIP solved using CPLEX.



(b) The effect of varying the tolerance on ISG solution time.



(c) Impact of interval uncertainty on solution quality and robustness

Figure 2: Runtime and solution quality analysis for ISG.

teed payoff is the payoff that the ISG method is able to guarantee against any rational attacker with payoffs that lie within the given intervals.

As we see in the results, there is a small decrease in the payoffs for the solutions to the interval games against the Nash and Stackelberg attackers. This is expected, and can be interpreted as the price of robustness, or the price of uncertainty. The advantage of the ISG method comes in the Guaranteed and Worst Case payoffs. There is an increasing trend in the worst-case payoffs for ISG, with the strongest results for very large intervals. More importantly, the method is able to guarantee high payoffs for smaller amounts of possible variation in the attackers payoffs, anywhere within the specified intervals.

7.2 Distributional Game Experiments

Our next set of experimental results evaluates the potential for ISG to be used as a fast approximation algorithm for distributional security games. We compare the performance of ISG using our methodology for transforming distributional security games into approximate versions based on intervals to the best existing methods for DSG, Greedy Monte Carlo (GMC) [12] and BRASS [17]. We run experiments on the same three classes of distributional games used by Kiekintveld et al., games with Uniform distributions of payoffs, games with Gaussian distributions with the same standard deviation for every payoff, and games with Gaussian distributions with varying standard deviations.

We use 300 random instances of each of these classes of games for the experiments in this section. The games are generated by first drawing random rewards and penalties for both players. All rewards are drawn from U[6; 8] and penalties are drawn from U[2; 4]. We then generate distributions of the correct type for the attacker's payoff, using the values drawn in the first stage as the mean. In uniform games we vary the length of the uniform interval to increase or decrease uncertainty. For Gaussian games we vary the standard deviation, and all payoffs have the same amount of uncertainty. Gaussian variable games have a different standard deviation for each payoff distribution. These standard deviations are drawn from U[0; 1] in our experiments.

All three of the algorithms we test have parameters. For GMC the two main parameters that control the solution time and quality are the number of sample attacker types used in the calculation, and the size of the increment used in the greedy allocation of coverage probability. Solution quality improves with a larger number of types and a smaller increment, but solution time increases. We include both a "low" and "high" quality set of parameters for GMC; the specific settings are given in Table 1. The parameter for BRASS is ϵ, and reflects how far attackers may be from choosing the optimal target. The parameters for ISG are the multiplier used to generate the interval game from the distributional game, and the tolerance. The tolerance does not have a large effect on the solution quality (because we can always get very small error), so we fix this at 0.0001 in our experiments.

Table 1: Parameter settings for the algorithms.

Parameter	Values
ISG Multipliers	0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0
BRASS Epsilons	0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0
GMC Low	increment 0.05,1000 types
GMC High	increment 0.01,10000 types

The BRASS epsilon parameter and ISG multiplier can both have a significant effect on the solution quality, and there is no obvious way to set the correct value of these parameters. The best values can depend on the size of the game, the type of uncertainty, and the amount of uncertainty (e.g., the standard deviation of the Gaussian distributions). In general, we test a variety of parameter settings and select the one that gives the best result. The set of candidate values for each algorithm is shown in Table 1. Figure 3(g) shows an example of an experiment to find the best parameters games with Gaussian uncertainty for the ISG algorithm. The amount of uncertainty (i.e., standard deviation) varies along the x-axis, the value of the best parameter setting is given on the y-axis, and each line represents a class of games with a different number of targets. In general, the best multipliers are smaller for smaller games and games with greater uncertainty. The need to do some experimentation to find a good parameter setting for ISG is not a large concern in practice. These settings could be used based on known values for similar classes of games, and the algorithm is fast enough that even testing a few different parameter setting on a specific game to find the best result would not be prohibitive. In all of our experiments, we tested all of the values in the table for both BRASS and ISG, and selected the one that gave the best result for the specific setting.

The first three plots, 3(a), 3(b), and 3(c) compare the solution qualities achieve by the algorithms on relatively small games with 15 targets and 3 resources. For the Uniform and Gaussian games we vary the amount of payoff uncertainty on the x-axis by varying the standard deviations of the attacker's payoff distributions. For Gaussian Variable games we use only a single setting, since the amount of uncertainty varies on a per-payoff basis in these games. In all cases, the defender's expected payoff for the solution is plotted on the y-axis. This is evaluated after the algorithms return solutions by using a very large number of Monte-Carlo sample types (100000) to give a very accurate estimate of the expected payoff for the proposed coverage solution. We also include a final baseline called "Mean" that solves the game optimally by assuming that the mean of the distribution is the exact payoff value (in other words, it ignores the uncertainty in payoffs and solves it as a standard security game).

The mean baseline performs very poorly in all cases. For uniform games, ISG has the highest solution quality, followed closely by BRASS. For Gaussian and Gaussian variable games ISG performs slightly worse than the GMC methods, particularly when there is a large amount of uncertainty. However, the performance is still very competitive.

In the second set of three plots, 3(d), 3(e), and 3(f), we fix the amount of uncertainty for each the three classes of games and vary the number of targets to show the performance on larger games. The standard deviation for uniform and Gaussian games is fixed at 0.5, while the Gaussian variable games use the same distribution of standard deviations as before. BRASS is not included in this set of results because it was too slow and required too much memory to complete for some of the larger problems. The pattern of results is similar to the smaller games. In uniform games there is a greater separation between the algorithms, with ISG outperforming GMC. On Gaussian and Gaussian variable games, GMC has higher solution quality, but the overall difference between GMC and ISG is fairly small.

The final result is presented in Figure 3(h). This plot compares the runtimes for computing solutions on the large Gaussian games (results for the other classes of games are very similar. Here we see that the solution times for both BRASS and GMC high rapidly increase as the size of the game increases. The solution times for GMC low and mean grow more reasonably. However, ISG is by far the fastest algorithm. It is fast enough to scale well beyond 100 targets, as seen in the previous set of results. Overall, ISG offers very fast solutions and either superior or competitive solution quality for approximating distributional games, depending on the type of uncertainty. It is a particularly good choice for accounting for uncertainty in very large games or situations where very fast performance is needed; in these cases it may be the only feasible method from a computational perspective that can account for uncertainty.

8. CONCLUSION

Security games have important real-world applications, but one of the critical questions is how to account for uncertainty and error in building the analyzing the game models. If the solutions are not robust to the kinds of errors and uncertainties that arise in modeling real problems, then they will not be useful in many situations. However, many of the standard approaches for handling uncertainty, such as Bayesian games, are very challenging from both a model elicitation standpoint and a computational standpoint.

We have introduced a new model of security games with uncertainty that is based on using intervals to represent possible payoffs, and takes a worst-case approach to uncertainty. This approach is motivated in part by the literature on robust optimization and more recently work on robust game theory concepts. We show that modeling uncertainty using intervals has distinct computational advantages. We present a highly efficient polynomial algorithm for approximating solutions to interval security games within very small (negligible) error bounds. Our experiment results show that this algorithm is much faster than equivalent MIP formulations.

In addition, we show that intervals can be used as a way to approximate infinite Bayesian games with distributional uncertainty. We develop a methodology for modeling the infinite games using games with intervals, which can then be solved using our fast algorithm for the interval case. In our experimental results, the solutions found using this interval-based approach are surprisingly good–in all cases they are competitive with the best known methods for directly approximating the solutions to the Bayesian games, and in some cases the quality is even better. In addition, the speed of the solutions is much faster, and we can scale to extremely large games using this approach. This provides a computationally feasible way to account for uncertainty even in very challenging cases of security games. Our success in applying interval-based models in this case also raises many interesting questions for future work in applying these basic principles to manage uncertainty in other, more general classes of games.

9. REFERENCES

[1] M. Aghassi and D. Bertsimas. Robust game theory. *Mathematical Programming*, 107:231–273, 2006. 10.1007/s10107-005-0686-0.

[2] N. Agmon, S. Kraus, G. A. Kaminka, and V. Sadov. Adversarial uncertainty in multi-robot patrol. In *IJCAI-09*, 2009.

[3] T. Alpcan and T. Basar. A game theoretic approach to decision and analysis in network intrusion detection. In *Proc. of the 42nd IEEE Conference on Decision and Control*, pages 2595–2600, 2003.

[4] N. Basiloco, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, 2009.

[5] A. Ben-Tal and A. Nemirovski. Robust optimization – methodology and applications. *Mathematical Programming*, 92:453–480, 2002. 10.1007/s101070100286.

[6] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *ACM EC-06*, pages 82–90, 2006.

[7] N. Gatti. Game theoretical insights in strategic patrolling: Model and algorithm in normal-form. In *ECAI-08*, pages 403–407, 2008.

[8] J. C. Harsanyi. Games with incomplete information played by Bayesian players (parts i–iii). *Management Science*, 14, 1967–8.

[9] M. Jain, E. Kardes, C. Kiekintveld, M. Tambe, and F. Ordonez. Security games with arbitrary schedules: A branch and price approach. In *AAAI-10*, 2010.

[10] M. Jain, M. Tambe, and C. Kiekintveld. Quality-bounded solutions for finite bayesian stackelberg games: Scaling up. In *AAMAS-11*, 2011.

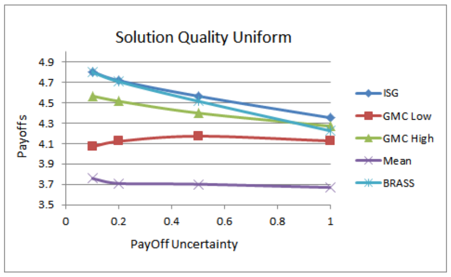
[11] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordonez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS-09*, 2009.

[12] C. Kiekintveld, J. Marecki, and M. Tambe. Approximation methods for infinite bayesian stackelberg games: Modeling distributional payoff uncertainty. In *AAMAS-11*, 2011.

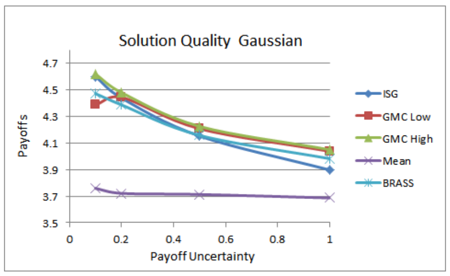
[13] D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, , and M. Tambe. Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of AI Research (JAIR) (to appear)*, 2011.

[14] O. Lerma, V. Kreinovich, and C. Kiekintveld. Linear-time resource allocation in security games with identical fully protective resources. In *Proceedings of the AAAI Workshop on Applied Adversarial Reasoning and Risk Modeling (AARM)*, 2011.

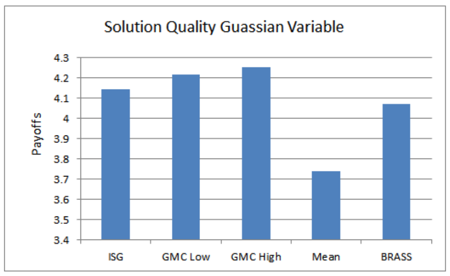
[15] K. C. Nguyen and T. A. T. Basar. Security games with incomplete information. In *Proc. of IEEE Intl. Conf. on Communications (ICC 2009)*, 2009.



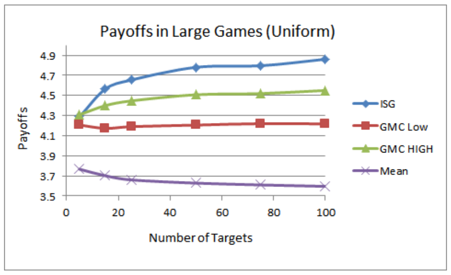
(a) Solution quality results for small games with uniform attacker payoff distributions.



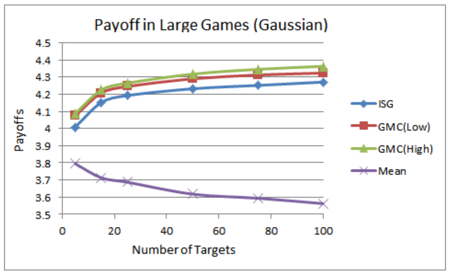
(b) Solution quality results for small games with Gaussian attacker payoff distributions.



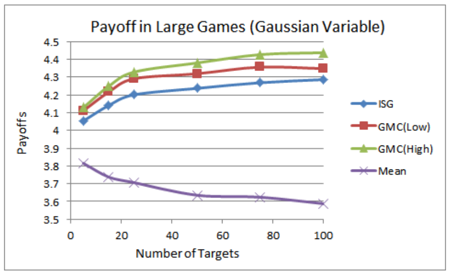
(c) Solution quality results for small games with Gaussian variable attacker payoff distributions.



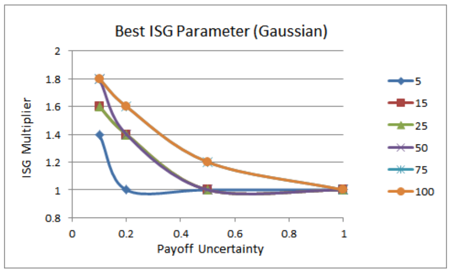
(d) Solution quality results scaling to larger games for uniform attacker payoff distributions.



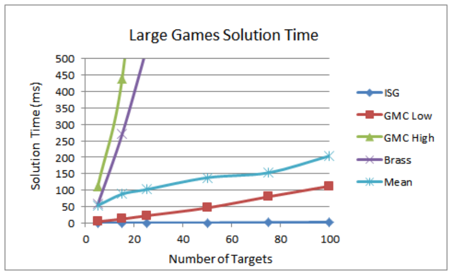
(e) Solution quality results scaling to larger games for Gaussian attacker payoff distributions.



(f) Solution quality results scaling to larger games for Gaussian variable attacker payoff distributions.



(g) Parameter optimization for ISG on games with Gaussian distributions.



(h) Comparison of solution times for solving large Gaussian games.

Figure 3: Solution quality and computation time comparisons.

[16] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *AAMAS-08*, pages 895–902, 2008.

[17] J. Pita, M. Jain, F. Ordóñez, M. Tambe, S. Kraus, and R. Magori-Cohen. Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *AAMAS-09*, 2009.

[18] J. Pita, M. Jain, C. Western, C. Portway, M. Tambe, F. Ordonez, S. Kraus, and P. Parachuri. Depoloyed ARMOR protection: The application of a game-theoretic model for security at the Los Angeles International Airport. In *AAMAS-08 (Industry Track)*, 2008.

[19] J. Pita, M. Tambe, C. Kiekintveld, S. Cullen, and E. Steigerwald. Guards - game theoretic security allocation on a national scale. In *AAMAS-11 (Industry Track)*, 2011.

[20] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. Direnzo, G. Meyer, C. W. Baldwin, B. J. Maule, and G. R. Meyer. PROTECT : A Deployed Game Theoretic System to Protect the Ports of the United States. *AAMAS*, 2012.

[21] M. Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

[22] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordóñez, and M. Tambe. IRIS - A tools for strategic security allocation in transportation networks. In *AAMAS-09 (Industry Track)*, 2009.

[23] H. von Stackelberg. *Marktform und Gleichgewicht*. Springer, Vienna, 1934.

[24] K. wei Lye and J. M. Wing. Game strategies in network security. *International Journal of Information Security*, 4(1–2):71–86, 2005.

[25] Z. Yin and M. Tambe. A unified method for handling discrete and continuous uncertainty in bayesian stackelberg games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012.